university of groningen / faculty of science and engineering

Multivariate Time Series Classification Using Conceptors: Exploring Methods Using Astronomical Object Data

Bachelor's Project Thesis

Jamie Vlegels, s3805719, j.vlegels@student.rug.nl,
Supervisor: Prof Dr H. Jaeger

Abstract: Time series classification tasks can be found in a wide range of real-world application domains, ranging from human activity recognition to electrocardiogram diagnostics. Echo state networks have already proven to be excellent tools for such tasks, providing a natural expression of temporal dynamics within their recurrent neural connections. Conceptors, neuro-computational mechanisms, extend the scene of recurrent neural networks by providing high-level conceptual and logical control on the low-level dynamics of a network. During their conception, a demonstration displayed state-of-the-art performance on a time series classification task using a conceptor classification scheme. This paper builds upon this demonstration to further explore conceptors for time series classification. To do this, five variations to the original conceptor classifier were proposed and evaluated on an astronomical object benchmark dataset. Additionally, an analysis of the performance of a heuristic for determining the value of a key conceptor parameter, aperture, was provided. It was found that all of the proposed conceptor classifier variations and baseline echo state network classifiers were able to achieve top-level performance on the benchmark dataset compared to previous methods applied to this dataset. Furthermore, the aperture analysis has shown that a significant increase in conceptor classifier performance can be achieved when choosing the value of the aperture parameter carefully. These findings provide additional evidence that conceptors can be utilised effectively for time series classification tasks.

Contents

1 Introduction

Time is a constituent of everything observable and a fundamental component of the human experience and the dynamics of real-world processes. As a result, countless processes are registered with some notion of time. Their dynamics can be captured as a collection of ordered, repeated measurements of the system over time, a *time series*. These measurements can consist of either single or multiple time-dependent variables, defining univariate and multivariate time series, respectively.

A challenging task involving time series concerns time series classification (TSC), where the presented series are assumed to belong to one of finitely many predefined but unknown classes, and the goal is to assign them this corresponding class label. This is a fundamental task for a broad range of real-world applications, ranging from human activity recognition to electrocardiogram diagnostics.

Recurrent neural networks (RNNs) seem to be appealing tools for such time series processing tasks. They belong to a class of artificial neural networks where abstract neurons form a recurrent connection between them, thus allowing them to express temporal dynamics. It can be shown that they are, under some assumptions, universal approximators of dynamical systems (Jaeger, 2001).

However, while various algorithms have been developed within this class, they often suffer from problems when trained via gradient descent due to slow convergence, vanishing or exploding gradients, or disruption when the network is driven through bifurcations (Lukoševičius & Jaeger, 2009). Because of this, Jaeger (2001) took a new approach to RNN design under the name of Echo State Networks (ESNs), which belongs to a collective trend referred to as Reservoir Computing (RC). This approach overcomes the aforementioned shortcomings of gradient descent RNN training by keeping the connection weights from the input layer to the recurrent layer (reservoir) and the connections within this reservoir fixed at their random initialisation; no further adaptation of these weights is undertaken. The desired output can then be produced by a trainable linear combination of all of these response signals. Consequently, ESNs have shown promising results on various TSC tasks such as speech recognition (Skowronski & Harris, 2006), (Skowronski & Harris, 2007) with low computational burdens.

Broadening the RNN scheme, Jaeger (2014) proposed neuro-computational mechanisms, called *conceptors*, which are able to control the neural dynamics conceived within RNNs using principles of conceptual abstraction and logic. They act as high-level neural filters for the low-level activation within a network and can be invoked to constrain the neural state space to the shape that was induced by patterns from which the conceptors were learnt. The strength at which these conceptors constrain neural activation is dictated by the aperture parameter and is of great importance for achieving proper filtering.

Conceptors also allow for a discriminative and conceptual classification scheme, where they are used as prototypes to conceptualise the various classes of patterns, on which a similarity measure with a novel pattern can be imposed to classify it. In Jaeger (2014), state-of-the-art performance of a conceptor classifier built upon an underlying ESN was achieved on a common benchmark, the *japanese vowel* recognition task. These favourable characteristics of ESNs and conceptors, together with state-of-the-art performance on the aforementioned benchmark dataset, indicate merit for the TSC scene.

Additionally, this proposed classification scheme was, in Jaeger's own words: "based on numerous ad-hoc design decisions and should be regarded as no more than a first demonstration of the basic usefulness of conceptors for classification tasks" (Jaeger, 2014). This, for example, includes a useful heuristic for determining the value of the aforementioned aperture parameter. This statement, indicating much room for various alterations to the classification scheme to be explored, became the main motivator behind this research, for which I have two aims:

i Propose variations to the conceptor classifier presented by Jaeger (2014) for TSC and analyse their performance.

ii Analyse the performance of the aperture heuristic.

The first aim will be achieved by proposing five conceptor classifier variations, built upon ESNs, and evaluating their performance with respect to the conceptor classifier by Jaeger (2014) and baseline ESN classifiers on an astronomical object

benchmark dataset. Additionally, the performance evaluations of various previous methods applied to this dataset will further assist in this evaluation. The second will be achieved by comparing the performance of conceptor classifiers utilising the aperture heuristic to the performance of conceptor classifiers for which their aperture values were optimised by an iterative optimisation scheme.

2 Background

In this section, the background necessary for this research is laid out. This starts with a description of the underlying echo state networks (ESNs) used for the conceptor classifiers. It will also include two common ESN methods for classifying time series, to function as baseline models for a final performance evaluation. Thereafter, the general workings of conceptors are explained, which will include a description of how to compute them, as well as some of their properties and functionalities. Finally, a summary of the previous methods applied to the dataset used in this study, as presented in Ruiz, Flynn, Large, Middlehurst, and Bagnall (2021) is given. The technicalities of the conceptor classifier as given in the demonstration by Jaeger (2014) will be provided in the methods section, to be expanded upon in this thesis.

2.1 Echo State Networks

An ESN, as introduced in Jaeger (2001) and Jaeger, Lukoševičius, Popovici, and Siewert (2007), is a dynamical system comprised of a reservoir; a recurrent layer of size N_x, to which N_u input and N_y output units are added. Here, only the connections from the reservoir to the output layer are learnt, the connections from the input layer and within the reservoir remain at their random initialisation.

The network provides a nonlinear expansion of the input signal fed to it, where the goal is to obtain an information-rich activation state space in $x(n)$ $(n = 1, \ldots, T)$ to be able to obtain the desired output y^{target} from a linear combination of it (Lukoševičius, 2012). Here, T is the length of the signal in discrete time. The network also acts as a form of memory, as activation values can propagate through the network via its recurrent connections as n increases.

The system can be described by the update equations

$$\tilde{x}(n+1) = \tanh\left(Wx(n) + W^{\text{in}}u(n+1) + b\right), \tag{2.1}$$

$$x(n+1) = (1-a)\,x(n) + a\tilde{x}(n+1), \tag{2.2}$$

where u and x are the input signal and internal activation states, W^{in} and W are the $N_x \times N_u$ and $N_x \times N_x$ weight matrices for the input and reservoir connections, and b is the bias vector, respectively.

Then, given some $N_y \times N_x$ matrix W^{out} computed from the internal activation states after running the network, the output of timestep n is given by

$$y(n) = W^{\text{out}}x(n). \tag{2.3}$$

2.1.1 Classification With One-hot Label

Using an ESN for TSC usually involves learning from class labels represented by a one-hot encoded vector, where the entry corresponding to the correct class is equal to one and zero for the other entries. The goal is then to find W^{out} which transforms the obtained (time-averaged) activation states from the network to approximate this representation.

In independent sessions starting from a fixed starting activation state vector $x(0) = x_{\text{start}}$, this is done by driving the network with samples $u^k(n)$ $(k = 1, \ldots, k_{\max};\ n = 1, \ldots, T)$, where k is the sample index and $k_{\max}$ is the number of samples in the training set, which includes the samples of all classes. Here, a sample is assumed to be an entire sequence corresponding to one observation with fixed length T, which will be standard throughout this thesis. Then, having obtained the network activation states $x^k(1), \ldots, x^k(T)$ $(k = 1, \ldots, k_{\max})$, the N_x sized vectors Σx^k are obtained by taking the average value of the activation states for that sample over time

$$\Sigma x^k = \frac{1}{T}\sum_{n=1}^{T} x^k(n), \tag{2.4}$$

as recommended in Lukoševičius (2012). This will result in a time-averaged activation state vector for every sample.

These vectors Σx^k are then horizontally concatenated into a $N_x \times k_{\max}$ state collection matrix $X = [\Sigma x^1; \ldots; \Sigma x^{k_{\max}}]$. Correspondingly, a

$N_y \times k_{\max}$ target matrix $Y^{\text{target}} = [y^1; \ldots; y^{k_{\max}}]$ is constructed from the $k_{\max}$ one-hot encoded class label vectors y^k $(k = 1, \ldots, k_{\max})$, where N_y is the number of dimensions of this one-hot representation and thus classes to be distinguished. Here, for every k, the entry of the k-th vector in Y^{target} corresponding to the true class label of the matching k-th vector of state collection matrix X is one and zero for the other entries.

Then, computing W^{out} is usually done via Ridge regression, also known as regression with Tikhonov regularisation

$$W^{\text{out}} = Y^{\text{target}} W^\top \left(XX^\top + \beta I\right)^{-1}, \qquad (2.5)$$

where β is a regularisation coefficient and I is the identity matrix.

Classifying a sample u from a testing set means feeding it to the network and obtaining the activation states x from the network's response. These are then again time-averaged into a vector Σx and then multiplied with W^{out} to obtain y

$$y = W^{\text{out}} \Sigma x. \qquad (2.6)$$

The predicted class label j^* for sample u is then decided by

$$j^* = \underset{j}{\operatorname{argmax}}\ y_j, \qquad (2.7)$$

where y_j is the j-th entry of y.

2.1.2 Classification With One-step Prediction

Another common method to utilise ESNs for TSC is to train them as 1-step predictors of the input signal. This is done by computing an output weight matrix per class by learning on time-delayed versions of the input signals fed to the reservoir. Classifying a sequence would then mean predicting this signal using the generated output weights and assigning the class label for which the output weights predicted the signal with the lowest error, defined by some error function.

Technically, for each class j, an output weight matrix W_j^{out} is learnt from the m_j class training samples $u_j^k(n)$ $(j = 1, \ldots, j_{\max};\ k = 1, \ldots, m_j;\ n = 1, \ldots, T)$, where $j_{\max}$ is the number of classes, as follows:

- In m_j independent sessions starting from $x(0) = x_{\text{start}}$, a network is driven with the input of each training sample u_j^k $(k = 1, \ldots, m_j)$. Then, the network activation states $x^k(1), \ldots, x^k(T-1)$ are extracted. The last timestep is pruned as a target in the input signal with a timestep greater than T does not exist.
- All of these activation states x^k are then horizontally concatenated into a $N_x \times ((T-1) \cdot m_j)$ matrix $\mathrm{X} = [x^1(1), x^1(2), \ldots, x^{m_j}(T-2), x^{m_j}(T-1)]$.
- Additionally, the sample input vectors u_j^k are horizontally concatenated into a $N_{\text{in}} \times ((T-1) \cdot m_j)$ matrix $\mathrm{Y}^{\text{target}} = [u_j^1(2), u_j^1(3), \ldots, u_j^{m_j}(T-1), u_j^{m_j}(T)]$. These input vectors represent the target prediction; the first timestep of each sample is therefore pruned.
- The output weights W_j^{out} are therafter computed according to Equation 2.5.

Then, classifying a sample u from the testing set means feeding it to the network and, together with the output weights W_j^{out} $(j = 1, \ldots, j_{\max})$, obtaining the output vectors $y_j(n)$ $(n = 1, \ldots, T)$ from the networks response with 2.3. The last timestep of the output vectors and the first timestep of the input vector are subsequently pruned to match the predictions with the corresponding input entry.

Finally, the predicted class label j^* for test sample u is then decided by picking j for which the Root-Mean-Squared error is minimal:

$$j^* = \underset{j}{\operatorname{argmin}} \sqrt{\frac{1}{T-1} \sum_{n=1}^{T-1} \left(u(n+1) - y_j(n)\right)^2}. \qquad (2.8)$$

2.2 Conceptors

When a recurrent neural network such as the one described in section 2.1 is driven by some signal u, the resulting activation states x from its update equations can be thought of to lie in an N_x dimensional state cloud. From this, a conceptor C can be created which, geometrically, takes on the form of an ellipsoid, of which its main axes represent the principal components of this state cloud.

Such a conceptor can be thought of as a neural filter which characterises the activation patterns within the network driven by u. For my purposes, this conceptor C is an $N_x \times N_x$ soft projection matrix, projecting activation states to a subspace spanned by several principal components of the state cloud.

Conceptor matrices can be obtained by minimising the loss function

$$\mathcal{L}(C) = E\left[\|Cx - x\|^2\right] + \alpha^{-2}\|C\|^2_{fro}, \tag{2.9}$$

where the expectation E is taken over all states x.

From this equation, it can be observed that a balance is struck between suppressing all and none of the leading components of the conceptor. This balance is dictated by the aperture parameter α, where small values will shape the conceptor to the null matrix (suppressing all states) and large values to the identity matrix (no suppression).

Minimising the loss $\mathcal{L}(C)$ leads to the analytical solution

$$C = R\left(R + \alpha^{-2} I\right)^{-1}, \tag{2.10}$$

where $R = E\left[xx^\top\right]$ is an $N_x \times N_x$ correlation matrix with the expectation E is taken over activation states x and I is the identity matrix. Practically, R is estimated with the approximation $R = XX^\top/T$, where $X = [x(1), \ldots, x(T)]$ is the $N_x \times T$ state collection matrix from activation states x and T is again the length of the signal.

2.2.1 Boolean Logic

A key characteristic of conceptors is that they allow for the logic-based operations OR, NOT, and AND to be applied to them, for which almost all laws of Boolean logic hold. On the data level, the OR ($\vee$) operation can be semantically interpreted as the merge of two datasets. For example, the resulting conceptor from the OR operation of two separate conceptors C and B can be described by the conceptor that would be derived from the union of the two activation state sets from which C and B were computed. This operation applied to conceptors C and B is formalised as

$$C \vee B = \neg\left(\neg C \wedge \neg B\right). \tag{2.11}$$

The NOT ($\neg$) operation can be interpreted as the inversion of the principle component weights of a dataset and its application to conceptor C is formalised as

$$\neg C = I - C. \tag{2.12}$$

The AND ($\wedge$) operation can then be straightforwardly interpreted by considering the interpretations of the OR and NOT operations together with de Morgan's rule. This operation applied to the conceptors C and B is then given as

$$C \wedge B = \left(C^{-1} + B^{-1} - I\right)^{-1}. \tag{2.13}$$

2.3 Previous Methods

The multivariate time series classification benchmark by Ruiz et al. (2021) will be used to obtain the performance evaluations of sixteen classifiers which were applied to the dataset used in this research, discussed in detail in section 3.1. This subsection is devoted to briefly describe the aforementioned classifiers.

First, three variations on the dynamic time warping (DTW) algorithm are considered, proposed by Shokoohi-Yekta, Hu, Jin, Wang, and Keogh (2017). Dynamic time warping is a 1-nearest neighbours classifier with a distance metric accounting for some offset between the compared time series. Independent DTW ($\mathrm{DTW_I}$) treats each dimension of the multivariate time series separately. Dependent DTW ($\mathrm{DTW_D}$) measures distances between multiple dimensions simultaneously. Adaptive DTW ($\mathrm{DTW_A}$) implements a threshold to decide whether to use $\mathrm{DTW_I}$ or $\mathrm{DTW_D}$.

The second classifier described in their paper is the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) (A. Bagnall, Flynn, Large, Lines, & Middlehurst, 2020) and is an ensemble of various univariate time series classifiers.

Other ensemble classifiers include Generalized random shapelet forest (gRFS) (Karlsson, Papapetrou, & Boström, 2016), Canonical interval forest (CIF) (Middlehurst, Large, & Bagnall, 2020), and The multiple representation sequence learner (MrSEQL) (Le Nguyen, Gsponer, Ilie, O'Reilly, & Ifrim, 2019). gRFS and CIF are tree-based classifiers using shapelets (Ye & Keogh, 2011) and Canonical Time-Series Characteristics, Catch22 (Lubba et al., 2019), respectively. MrSEQL is also a tree-based classifier which uses symbols derived

from Symbolic Aggregate Approximation (SAX) (Lin, Keogh, Wei, & Lonardi, 2007) or Symbolic Fourier Approximation (SFA) (Schäfer & Högqvist, 2012).

Another symbolic classifier WEASEL+MUSE (MUSE) (Schäfer & Leser, 2017) extracts words from the time series using Multivariate Unsupervised Symbols and Derivatives, MUSE and SFA. These words are filtered and the remaining words are used to build a logistic regression classifier.

The study also evaluates several deep learning-based methods, which includes Residual Network (ResNet) (Z. Wang, Yan, & Oates, 2017), InceptionTime (Ismail Fawaz et al., 2020), and Time series Attentional Prototype Network (TapNet) (Zhang, Gao, Lin, & Lu, 2020), all being convolutional neural networks. ResNet is comprised of three convolutional layers to which residual connections are added that let the input bypass these layers. InceptionTime builds upon ResNet by adding "inception" modules, which first reduce the dimensionality of the time series to then apply convolutions of various sizes. TapNet combines the output of a Long Short-Term Memory network and convolutional layers using random subsets of the dimensions of the time series. The dimensionality of these feature representations is reduced and used as a prototype for each class, which are then used to match a test sample based on a distance metric.

As a final classifier, the random convolutional kernel transform (ROCKET) (Dempster, Petitjean, & Webb, 2020) is described. It produces features based on a large number of random convolution kernels, which are thereafter used to construct a linear classifier.

3 Methodology

In this section, the methods utilised for achieving the aims of this research are described. First, the dataset used for the evaluation of the conceptor classifiers is laid out, together with some preprocessing steps. Thereafter, a quick note on the underlying networks used for the conceptor and baseline classifiers, as well as the method for aperture selection is provided. Then, the conceptor classifier as demonstrated in Jaeger (2014), together with the proposed alternative conceptor classifiers are described. Following this section, the experimental setup taken to evaluate the performance of these classifiers is discussed. Finally, an overview of the experimental setup of the aperture analysis is given.

This project was implemented in Python 3.10.0, for which the code is available at `https://github.com/VlegelsJamie/Conceptor-Classifiers`.

3.1 Dataset

The performance of each conceptor classifier was evaluated on the *LSST* astronomical object dataset. The dataset represents a subset of the dataset used for The Photometric LSST Astronomical Time Series Classification Challenge (PLAsTiCC) (Bahmanyar et al., 2018) and was created by A. J. Bagnall et al. (2018) for multivariate time series classification performance benchmarking. The dataset is publicly available and was retrieved from their Time Series Classification website in the datasets listing `http://www.timeseriesclassification.com/dataset.php`.

Originating from a 2018 Kaggle competition, PLAsTiCC was an open data challenge in preparation for observations from the Large Synoptic Survey Telescope (LSST), which will gather information on a large portion of the sky over 10 years from 2022 onwards. The challenge entailed classifying light curves driven by the physical processes of 14 different classes of astronomical objects, ranging from supernovae to binary stars orbiting around their common barycenter. These light curves are measurements of an object's brightness as a function of time — by measuring the photon flux in six different astronomical filters or *passbands*. These include the ultra-violet, visual and infrared regions of the light spectrum. As opposed to containing additional meta-data as in the PLAsTiCC dataset, the LSST dataset only contains the aforementioned light curves.

Each sample is provided as a 6-dimensional time series made from the flux measurements of 6 different passbands over a period of 36 timesteps. Figure 3.1 depicts an exemplary light curve for each class.

These samples were chosen to be representative of the shape of the majority of class samples. However, it must be noted that the range of observed values of samples follows a positively skewed distribution.

The dataset consists of 4925 samples, of which 2459 and 2466 were used to construct the train-

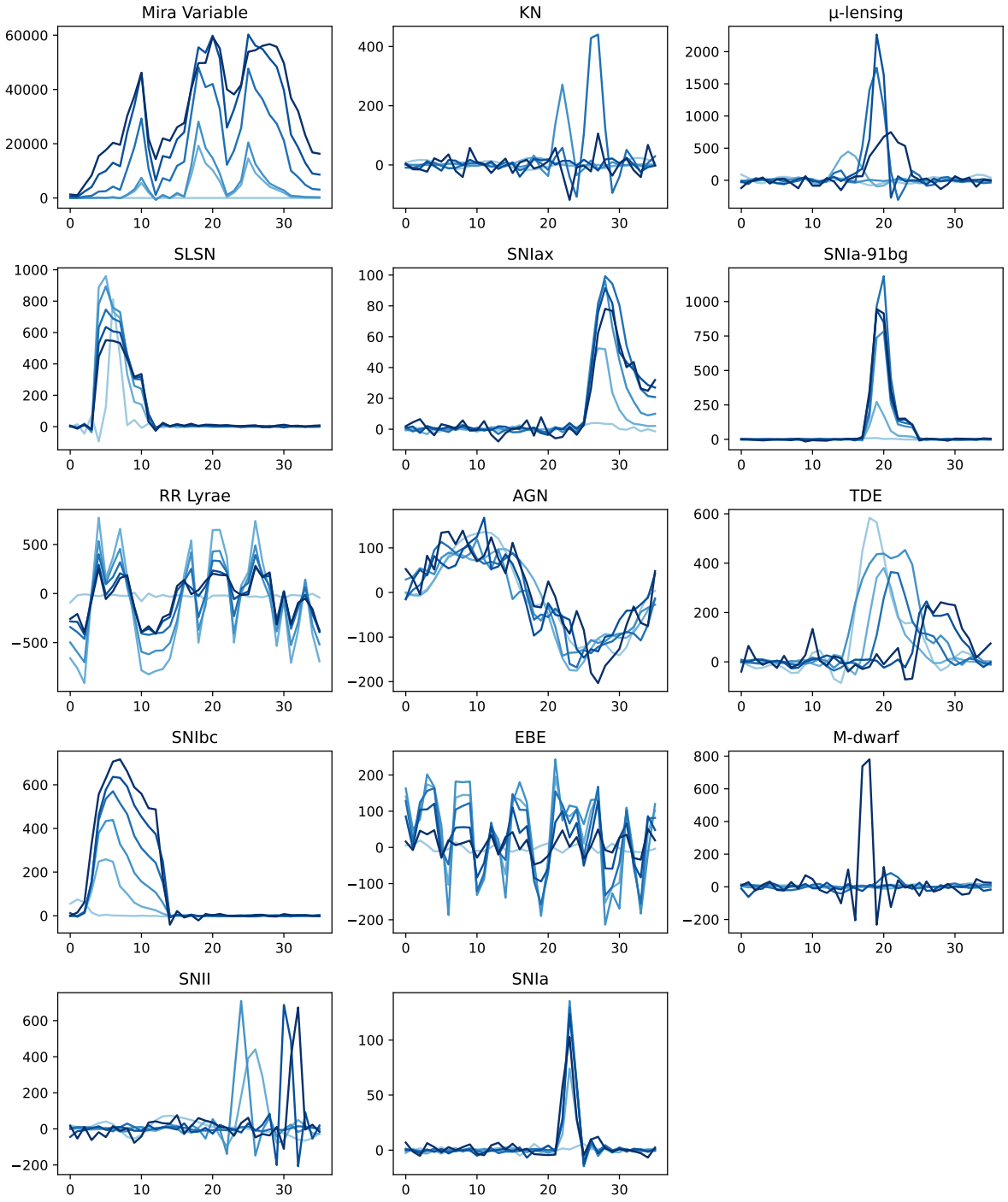

Figure 3.1: Exemplary light curves from the LSST dataset. Each subplot depicts a representative single class sample, in which the values of the 6 passbands are plotted against 36 discrete timesteps. Each colour represents a certain passband.

ing and testing set, respectively (per design of the benchmark). Table 3.1 shows the class distribution of the entire dataset, which can be observed to be imbalanced. The training and testing set both follow this presented class distribution, apart from seven classes for which the testing set has an additional sample.

Class Name	**#Samples**	**Samples** (%)
Mira Variable	14	0.28
KN	47	0.95
µ-lensing	69	1.40
SLSN	103	2.10
SNIax	125	2.54
SNIa-91bg	136	2.76
RR Lyrae	154	3.13
AGN	241	4.89
TDE	247	5.02
SNIbc	306	6.21
EBE	540	10.96
M-dwarf	626	12.71
SNII	763	15.49
SNIa	1554	31.55

Table 3.1: Distribution of class samples in the LSST dataset.

3.2 Preprocessing

As a first preprocessing step, the data is standardised to address the scale variance of the different passbands. This variance in scale might lead certain passbands with large scales to dominate the amount of impact on the classification decision compared to passbands with small scales.

To make each passband contribute approximately equally to the final classification decision, each passband was individually preprocessed into a standardised format. Due to the non-normality of the data (aforementioned positive skew), the median value and interquartile range from the passbands were used to shift and scale them respectfully. Formally, this means that for every passband i in the dataset, the transformed passbands $\tilde{S}_i$ were acquired according to

$$\tilde{S}_i = \frac{S_i - \mathbf{median}(\mathbf{S}_i)}{\mathbf{Q3}(\mathbf{S}_i) - \mathbf{Q1}(\mathbf{S}_i)}, \tag{3.1}$$

where S_i are the unprocessed values of passband i ($i = 0, \ldots, 6$), $\mathbf{median}(\mathbf{S}_i)$ is the median value of all unprocessed samples in the training set of passband i, and $\mathbf{Q3}(\mathbf{S}_i)$ and $\mathbf{Q1}(\mathbf{S}_i)$ are the values of the third and first quartile of all unprocessed samples in the training set of passband i.

As a second step, due to the risk of overexciting the network, the samples were then *individually* scaled an additional time such that the minimum and maximum values across the samples were confined to a range of $[-1, 1]$ according to

$$\bar{S}^k = \frac{\tilde{S}^k}{\mathbf{max}(\tilde{S}^k) - \mathbf{min}(\tilde{S}^k)}, \tag{3.2}$$

where $\bar{S}^k$ is the final preprocessed sample k, $\tilde{S}^k$ is the previously k-th processed sample, $\mathbf{max}(\tilde{S}^k)$ is the maximum value of sample $\tilde{S}^k$, and $\mathbf{min}(\tilde{S}^k)$ the minimum value of sample $\tilde{S}^k$.

Figure 3.2 displays the effect of the aforementioned preprocessing steps on samples from three different astronomical objects.

3.3 Network Configuration

For every conceptor and baseline classifier, an echo state network with 6 input units, a constant bias term, and leaky integrator units with update equations 2.1 and 2.2 was created.

This was done by creating a fully connected reservoir weight matrix W, an input weight matrix W^{in}, and a bias vector b sampled from a normal distribution centred around 0.0 with a standard deviation of 1.0. Thereafter, W was scaled to a spectral radius ρ, W^{in} to $W^{\text{in}}_{\text{scale}}$, and b to b_{scale}. The start activation state vector x_{start} was initialised to the zero vector. A note on how these parameter values were chosen is given in section 3.6.

3.4 Aperture Adaptation

Optimal aperture values for conceptors were found by first computing a preliminary conceptor $\tilde{C}$ from a correlation matrix R according to

$$\tilde{C} = R\left(R + I\right)^{-1} \tag{3.3}$$

and then adapting the aperture via an *aperture adaptation* operation φ on the conceptor $\tilde{C}$ according to

$$\varphi\left(C, \gamma\right) = \tilde{C}\left(\tilde{C} + \gamma^{-2}\left(I - \tilde{C}\right)\right)^{-1}, \tag{3.4}$$

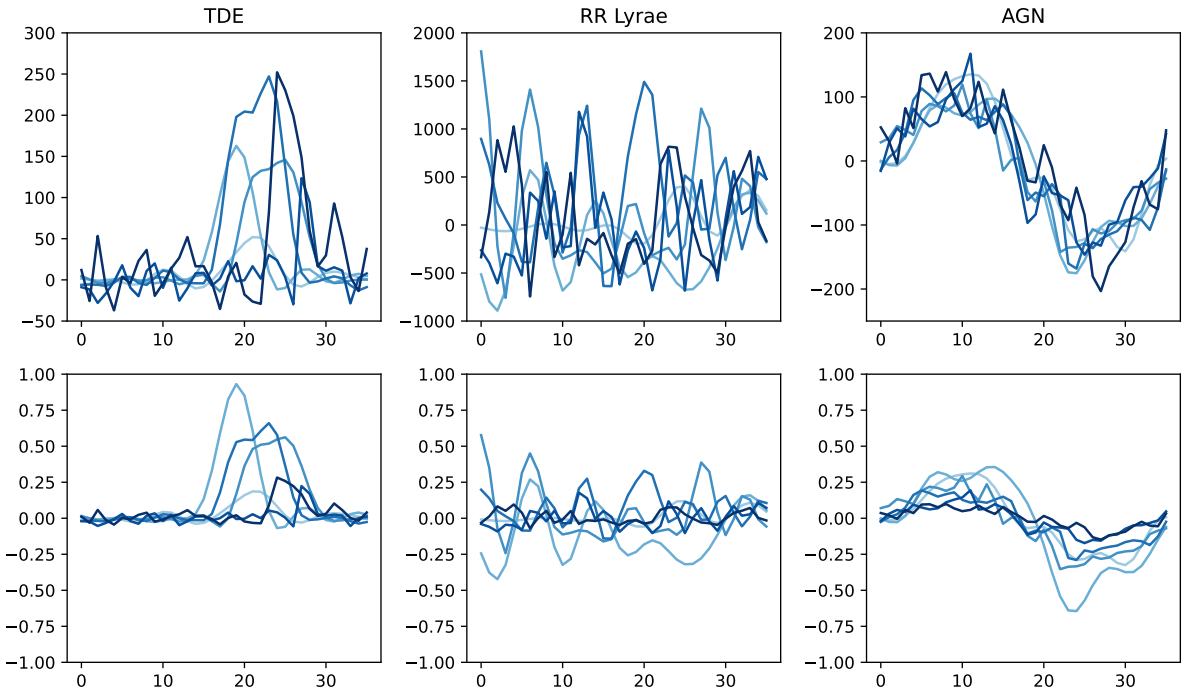

Figure 3.2: Three exemplary light curves from the LSST dataset before (top row) and after preprocessing (bottom row)

where $\varphi(C, \gamma)$ is the aperture adjusted conceptor obtained from the preliminary conceptor $\tilde{C}$ by scaling its aperture by a factor of γ.

Instead of cross-validation on the training data to optimise the aperture adaptation factor γ, Jaeger (2014) selects the value for which the gradient of the squared Frobenius norm

$$\Delta(\gamma) = \frac{d}{d\log(\gamma)} \| \varphi(C, \gamma) \|_{\text{fro}}^2 \tag{3.5}$$

is maximal. This research follows the same method for deciding on aperture values.

Practically, γ was obtained by computing the quantities $\|\varphi(\tilde{C}, 2^y)\|_{\text{fro}}^2$ for $y = 0, \ldots, 15$. Then, the pairwise differences between these values were computed. Finally, they were interpolated using a cubic spline and the support point $y_{\max}$ for which the value of the gradient was maximal was chosen for γ, yielding $\gamma = 2^{y_{\max}}$.

3.5 Conceptor Classifiers

In this section, first, the "standard" conceptor classifier as presented in Jaeger (2014) is laid out. Thereafter, all of the technicalities of the proposed conceptor classifiers will be presented. This includes three variations on the $\text{C}_{\text{Standard}}$ classifier: $\text{C}_{\text{Reduced}}$, $\text{C}_{\text{Combined}}$, and C_{Tubes}. These techniques primarily aim to provide an alternative way to employ the activation states from an ESN to construct conceptors. Additionally, two alternative conceptor classifiers based on reservoir excitation space similarity ($\text{C}_{\text{Reservoir}}$), and forecasting ($\text{C}_{\text{Forecast}}$) are proposed.

In contrast to the standard conceptor classifier, where an echo state network is utilised passively by merely acting as a feature expansion step, the latter two methods refer back to the network dynamics in the testing stage. In this way, a key characteristic of conceptors comes more actively into play, namely, acting as neural filters to confine a network to the mode of processing typical for the patterns the conceptor was learnt from.

3.5.1 $\mathbf{C_{Standard}}$

This classification method follows the demonstration in Jaeger (2014) and is presented here again for clarity.

Conceptually, this method involves learning a "positive evidence" conceptor for every class from the activation states induced in a network by the samples belonging to that class. Additionally, a corresponding "negative evidence" conceptor is computed, which represents the activation states that were not induced by any of the other class samples. When testing, the induced activation states of unseen samples are then directly compared to these conceptors, yielding evidence for both types of conceptors for each class. Finally, class labels are then decided based on the maximum value of the combination of both pieces of evidence.

Technically, for each class j, a positive evidence conceptor C_j^+ was first learnt from the m_j preprocessed class training samples $u_j^k(n)$ ($j = 1,\ldots,14;\ k = 1,\ldots,m_j;\ n = 1,\ldots,36$) as follows:

- In m_j independent sessions starting from $x(0) = x_{\text{start}}$, the network described in section 3.3 was run with the input of each training sample u_j^k ($k = 1,\ldots,m_j$). Then, the network activation states $x^k(1),\ldots,x^k(36)$ were extracted and concatenated into a single $36 \cdot N_x$ dimensional column-vector $\mathrm{x}_j^k = [x^k(1);\ldots;x^k(36)]$.
- The obtained x_j^k were then horizontally concatenated into a $(36 \cdot N_x) \times m_j$ matrix X from which a correlation matrix $R_j = \mathrm{X}\mathrm{X}^\top/m_j$ was computed. From R_j, a preliminary "positive evidence" conceptor $\tilde{C}_j^+$ was then constructed via Equation 3.3 and has a size of $(36 \cdot N_x) \times (36 \cdot N_x)$.

Thereafter, for each class j, an additional preliminary negative evidence conceptor $\tilde{C}_j^-$ which represents the event of not being any of the other classes was computed as

$$\tilde{C}_j^- = \neg \bigvee \{\tilde{C}_1^+,\ldots,\tilde{C}_{j-1}^+,\tilde{C}_{j+1}^+,\ldots,\tilde{C}_{14}^+\}. \quad (3.6)$$

Then, for both the preliminary positive evidence and negative evidence conceptors, their aperture was adapted using the method described in section 3.4. However, instead of adjusting the aperture by the mean values of the found aperture adaptation factors γ_j^+ and γ_j^- as in Jaeger (2014), the aperture adaptation factors remained unaltered once found for each conceptor. This has proven to yield a significant increase in performance for all of the conceptor classifiers. This adjustment will therefore be standard throughout this research.

Finally, classifying a sample u from the testing set meant feeding it to the network and again obtaining a single vector x consisting of the concatenated network activation states. For each positive evidence conceptor C_j^+, its corresponding negative evidence conceptor C_j^-, and test vector x, the positive evidence quantity E_j^+ was computed by

$$E_j^+ = \mathrm{x}^\top C_j^+ \mathrm{x}, \quad (3.7)$$

the negative evidence quantity E_j^- by

$$E_j^- = \mathrm{x}^\top C_j^- \mathrm{x}, \quad (3.8)$$

and the combined evidence quantity E_j by

$$E_j = E_j^+ + E_j^-. \quad (3.9)$$

The predicted class label j^* for sample u was then decided by

$$j^* = \operatorname*{argmax}_j E_j. \quad (3.10)$$

Following thorough experimentation, it was found that this equally weighted blend of positive and negative evidence did not always work optimally for the various classifiers proposed in this thesis. Large variations for both types of evidence were found depending on the parameters of the underlying network. To account for this type of flexibility, a linear blending operation of evidence

$$E_j = (1-\mu)E_j^- + \mu E_j^+ \quad (3.11)$$

is needed, where μ is a parameter indicating the amount of blending. $\mu = 1.0$ would mean only the positive evidence is used for E_j and $\mu = 0.0$ would mean only the negative evidence is used.

This is analogous (but less efficient to compute) to the morphing operation as proposed in Jaeger (2014)

$$C_j^* = (1-\mu)C_j^- + \mu C_j^+, \quad (3.12)$$

on the positive and negative evidence conceptors for $0.0 \leq \mu \leq 1.0$ and then computing the evidence value E_j directly from C_j^* with

$$E_j = \mathrm{x}^\top C_j^* \mathrm{x}. \quad (3.13)$$

For the remainder of this paper, this additional hyperparameter μ is introduced to the classifiers, which dictates this blending of evidence.

3.5.2 $\mathbf{C_{Reduced}}$

Due to the concatenation of activation states to form x, this vector scales according to the number of reservoir units N_x times the length of the sample, which is 36. Considering conceptors grow quadratically in size with x, this only allows underlying reservoirs with a small number of reservoir units to be employed for this task.

As a first method to reduce the dimensionality of the vectors from which the conceptors are constructed, the average of the activation states over time is taken instead of concatenating them together.

Technically, for each class j, a positive evidence conceptor C_j^+ was first learnt from the m_j preprocessed class training samples $u_j^k(n)$ ($j = 1,\dots,14;\ k = 1,\dots,m_j;\ n = 1,\dots,36$) as follows:

- In m_j independent sessions starting from $x(0) = x_{\text{start}}$, the network described in section 3.3 was run with the input of each training sample u_j^k ($k = 1,\dots,m_j$). Then, the network activation states $x^k(1),\dots,x^k(36)$ were extracted and time-averaged into a single N_x dimensional vector Σx^k according to 2.4.
- The obtained Σx^k were then horizontally concatenated into a $N_x \times m_j$ matrix X from which a correlation matrix $R_j = \mathrm{X}\mathrm{X}^\top/m_j$ was computed. From R_j, a preliminary "positive evidence" conceptor $\tilde{C}_j^+$ was then constructed via Equation 3.3 and has a size of $N_x \times N_x$.

The definitive positive and negative evidence conceptors C_j^+ and C_j^- were obtained according to the same procedure as with the standard conceptor classifier. These conceptors were then morphed to form C_j^* according to 3.12.

Classifying a sample u from the testing set meant obtaining a single vector Σx from the network's response, representing the time-averaged reservoir states. Then, the combined evidence value E_j was obtained by computing

$$E_j = \Sigma x^\top C_j^* \Sigma x \tag{3.14}$$

and the predicted class label j^* for sample u was decided by 3.10.

3.5.3 $\mathbf{C_{Combined}}$

As a second method, the activation states are left as is and function as separate feature vectors. Thus, each sample generates 36 feature vectors from which the conceptors will be learnt. These types of conceptors are standard throughout Jaeger (2014) for stationary sequences of potentially infinite length, but will now be tested on these non-stationary finite length sequences. The final evidence values will then be given as a summation of evidence over the timesteps.

More technically, for each class j, a positive evidence conceptor C_j^+ was again learnt from the m_j preprocessed class training samples $u_j^k(n)$ ($j = 1,\dots,14;\ k = 1,\dots,m_j;\ n = 1,\dots,36$) as follows:

- In m_j independent sessions starting from $x(0) = x_{\text{start}}$, the network described in section 3.3 was run with the input of each training sample u_j^k ($k = 1,\dots,m_j$). Then, the network activation states $x^k(1),\dots,x^k(36)$ were extracted. These activation states were left as is and not concatenated into a single column vector.
- The network activation states x^k were then horizontally concatenated into a $N_x \times (36 \cdot m_j)$ matrix $\mathrm{X} = [x^1(1), x^1(2), \dots, x^{m_j}(35), x^{m_j}(36)]$ from which a correlation matrix $R_j = \mathrm{X}\mathrm{X}^\top/(36 \cdot m_j)$ was obtained. From R_j, a preliminary positive evidence conceptor $\tilde{C}_j^+$ was then constructed again via Equation 3.3 and has a size of $N_x \times N_x$.

Definitive positive and negative evidence conceptors were then again computed from which C_j^* was formed.

Classifying a sample u from the testing set meant obtaining activation states x from the network's response. Then, the combined evidence value E_j was obtained by computing

$$E_j = \sum_{n=1}^{36} x(n)^\top C_j^* x(n) \tag{3.15}$$

and the predicted class label j^* for the sample u was decided by 3.10.

3.5.4 $\mathbf{C_{Tubes}}$

As a third method, conceptors are not only computed per class but also per timestep. Here, conceptors encode "state region tubes" along time characteristic of a particular class. The final evidence values will then be given as a summation of evidence over the timesteps, for which a separate conceptor is evaluated at each timestep.

Technically, for each class j, the positive evidence conceptors $C_j^+(n)$ $(n = 1, \ldots, 36)$ were learnt from the m_j preprocessed class training samples $u_j^k(n)$ $(j = 1, \ldots, 14;\ k = 1, \ldots, m_j;\ n = 1, \ldots, 36)$ as follows:

- In m_j independent sessions starting from $x(0) = x_{\text{start}}$, the network described in section 3.3 was run with the input of each training sample u_j^k $(k = 1, \ldots, m_j)$. Then, the network activation states $x^k(1), \ldots, x^k(36)$ were extracted.
- Then, for each timestep n, these activation states were horizontally concatenated into $N_x \times m_j$ matrices $\mathrm{X}(n) = [x^1(n), \ldots, x^{m_j}(n)]$ from which correlation matrices $R_j(n) = \mathrm{X}(n)\mathrm{X}^\top(n)/m_j$ were obtained. From $R_j(n)$, the preliminary positive evidence conceptors $\tilde{C}_j^+(n)$ were then again constructed via Equation 3.3 and have a size of $N_x \times N_x$.

Thereafter, for each class j and timestep n, the preliminary negative evidence conceptor $\tilde{C}_j^-(n)$ was computed via

$$\tilde{C}_j^-(n) = \neg \bigvee \{\tilde{C}_1^+(n), \ldots, \tilde{C}_{j-1}^+(n), \tilde{C}_{j+1}^+(n), \ldots, \tilde{C}_{14}^+(n)\} \tag{3.16}$$

and together with the positive evidence conceptor, their apertures were adjusted according to section 3.4. Then, for each class j and timestep n, the positive evidence conceptor $C_j^+(n)$ and negative evidence conceptor $C_j^-(n)$ were linearly interpolated to obtain $C_j^*(n)$ via

$$C_j^*(n) = (1 - \mu)C_j^-(n) + \mu C_j^+(n). \tag{3.17}$$

Classifying a sample u from the testing set meant again obtaining activation states x and computing the evidence quantity E_j by

$$E_j = \sum_{n=1}^{36} x(n)^\top C_j^*(n) x(n). \tag{3.18}$$

The predicted class label j^* for sample u was then decided by 3.10.

3.5.5 $\mathbf{C_{Reservoir}}$

As mentioned, this method utilises the characteristic of conceptors in their ability to filter activation states which are atypical to the patterns from which the conceptors were computed. With this method, the reservoir is run once to compute class conceptors and then a second time with the learnt conceptors in the update loop. In this second step, the activation states within the network are constrained by the applied conceptor. The class label is then assigned to the class for which its conceptor constrained these states the least, measured by the error between the original and constraint activation states. To apply a conceptor to the update loop of the ESN, the conceptors computed per timestep were utilised.

This means that, for each class j and timestep n, a positive evidence conceptor $C_j^+(n)$ was learnt as described in the second method of section 3.5.4. No negative evidence conceptors were computed for this method.

Then, classifying a sample u from the testing set meant feeding it to the network and obtaining the activation states x from the network's response.

Additionally, for each class j, the network was run in independent sessions starting from $x(0) = x_{\text{start}}$ with sample u, for which the positive evidence conceptors $C_j^+(n)$ $(j = 1, \ldots, 14;\ n = 1, \ldots, 36)$ were applied as

$$\hat{x}_j\,(n+1) = C_j^+(n)\; x\,(n+1)\,, \tag{3.19}$$

yielding the activation states $\hat{x}_j$.

The predicted class label j^* for sample u was then decided by picking j for which the Root-Mean-Squared error was minimal:

$$j^* = \operatorname*{argmin}_j \sqrt{\frac{1}{36}\sum_{n=1}^{36} \left(x(n) - \hat{x}_j(n)\right)^2}. \tag{3.20}$$

3.5.6 $\mathbf{C_{Forecast}}$

As a final method, not only are the internal reservoir activation states compared, but the network's response via its output units is also compared. This is done by doing a one-step prediction task.

This, firstly, entailed obtaining the output weight matrices W_j^{out} for each class j as described in section 2.1.2. Then, with the same obtained activation states, a positive evidence conceptor $C_j^+(n)$ was learnt for each class j and timestep n as described in the second method of section 3.5.4.

Classifying a sample u from the testing set meant obtaining the activation states x from the network's response. Additionally, the constrained activation states $\hat{x}_j$ were obtained via 3.19 using the conceptors $C_j^+(n)$ $(j = 1, \ldots, 14;\ n = 1, \ldots, 36)$.

As the first piece of evidence, the Root-Mean-Squared error was computed between input sample u and the obtained output values y_j from the one-step prediction method described in section 2.1.2, yielding an error value E_j^1 for every class j.

As the second piece of evidence, the Root-Mean-Squared error was also computed between input sample u and the obtained output values $\hat{y}_j$ from the same one-step prediction method using $\hat{x}_j$ as reservoir activation states, yielding an error value E_j^2 for every class j.

As the third piece of evidence, the Root-Mean-Squared error between outputs y_j and $\hat{y}_j$ was computed, yielding an error value E_j^3 for every class j.

As the last piece of evidence, the the Root-Mean-Squared error was computed between the activation states x and $\hat{x}_j$ as with the previous method, yielding an error value E_j^4 for every class j.

Finally, these error values were individually normalised to a range of $[0, 1]$ and the predicted class label j^* for sample u was then decided by picking j for which the sum of these errors was minimal:

$$j^* = \underset{j}{\operatorname{argmin}} \sum_{h=1}^{4} E_j^h. \tag{3.21}$$

3.6 Experimental Setup

All of the conceptor classifiers mentioned in section 3.5 were implemented and their performance was evaluated on the testing set. This performance was compared to the two ESN classifiers as described in sections 2.1.1 and 2.1.2, functioning as a baseline. These will be referred to as $\text{ESN}_{\text{One-hot}}$ and $\text{ESN}_{\text{Forecast}}$, respectively. It was also compared to the previous methods applied to this dataset, mentioned in section 2.3. Accuracy was used as a metric, defined as the ratio between the number of correct classifications and the number of samples in the testing set.

The echo state network parameters ρ, $W_{\text{scale}}^{\text{in}}$, b_{scale}, a, and N_x; morphing parameter μ; and Ridge regression regularisation coefficient β, were optimised for each conceptor and baseline classifier via Bayesian optimisation, evaluated using 5-fold stratified cross-validation over the training data (keeping class distribution equal across the folds). These evaluations were done using randomly initialised reservoirs to ensure robust parameter values that will keep the ESNs stable over a large set of random initialisations.

For this Bayesian optimisation scheme, the Matern kernel was used as a Gaussian process estimator with an added Gaussian noise with a variance of 0.04 to account for the performance difference of the randomly initialised reservoirs. The lower confidence bound was used as an acquisition function and was minimised by evaluating it over 10000 uniformly randomly sampled points and choosing the point for which the acquisition function was minimal as the next candidate minimum. This was repeated for 200 iterations.

Having obtained the optimal hyperparameters values, the classifiers were run for 40 trials with randomly initialised underlying reservoirs and evaluated on the testing set.

3.6.1 Aperture Optimisation

As the last objective of this research, the performance of the aperture adaptation heuristic 3.5 was analysed by comparing it to the performance of iteratively optimised aperture values. This was done by optimising the individual aperture values of the conceptors via the same Bayesian optimisation scheme as described previously, but without the added Gaussian noise to the estimator (no randomly initialised reservoirs were used; the weights remained fixed). Here, only the aperture values are optimised; the rest of the hyperparameters remain fixed at their previously optimised values. The apertures for the positive and negative evidence conceptors were optimised in separate runs and thereafter combined by searching for a new optimal μ. Due to computational constraints, this procedure was only employed for the $\text{C}_{\text{Standard}}$, $\text{C}_{\text{Reduced}}$, and $\text{C}_{\text{Combined}}$ classifiers.

To determine the space to search over, the con-

ceptor classifiers were run with a predetermined random seed and their heuristically determined aperture values were taken as the centre values of the search space dimensions. Then, for each positive aperture value, their upper and lower search space bounds were taken as the centre value plus and minus half of this value, respectively. It was found that the negative evidence conceptors allowed for significantly less leeway with respect to the range of "good" aperture values. This meant that the upper and lower search space bounds for the negative aperture values were taken as the centre value plus and minus ten percent of this value, respectively.

Once the optimal aperture values were found, the classifiers were run with the same random seed and evaluated on the testing set.

4 Results

This section will provide an overview of the achieved performance of the conceptor classifiers in comparison with the baseline classifiers and methods from previous work, as well as an overview of the aperture analysis results.

Table 4.1 shows the test accuracy (+ standard deviation) of the conceptor and baseline classifiers.

Method	**Acc** (%)
C_{Standard}	53.12±1.88
C_{Reduced}	66.84±1.34
C_{Combined}	65.93±1.41
C_{Tubes}	64.96±1.12
$C_{\text{Reservoir}}$	64.21±0.61
C_{Forecast}	65.63±0.80
$\mathbf{ESN_{One\text{-}hot}}$	**67.17±0.44**
ESN_{Forecast}	59.05±0.75

Table 4.1: Accuracy (+ standard deviation) of evaluated conceptor and baseline classifiers on the testing set averaged over 40 trials.

It can be seen that the ESN baseline classifier which discriminates class labels directly achieved the highest accuracy score. Of the conceptor classifiers, the C_{Reduced} classifier achieved the highest accuracy score. Additionally, it can be observed that, apart from the standard conceptor classifier, all conceptor classifiers performed better than the baseline ESN forecast classifier. The hyperparameters values that were used to obtain these results can be found in Appendix A

Figure 4.1 shows the accuracy score of the conceptor and the sixteen classifiers from previous work. It can be observed that the ESN baseline classifier $ESN_{\text{One-hot}}$ has the highest score, with five conceptor classifiers leading thereafter. C_{Standard} is the only classifier that performs worse than 50% of all classifiers.

4.1 Aperture Optimisation

Table 4.2 shows the test accuracy of the conceptor classifiers for which their aperture was determined via the aperture adaptation heuristic and for which their aperture was optimised via the Bayesian optimisation scheme.

	Acc (%)	
Method	**Heuristic**	**Optimised**
C_{Standard}	53.24	**54.14**
C_{Reduced}	66.10	**67.80**
C_{Combined}	61.39	**67.44**

Table 4.2: Accuracy of the evaluated conceptor classifiers with heuristically determined aperture values and Bayesian optimised aperture values on the testing set with a fixed initial reservoir.

It can be observed that for all classifiers, their accuracy increased when their aperture values were iteratively optimised. The most improvement in performance can be observed for the C_{Combined} classifier, for which its random seed resulted in relatively low performance using the aperture heuristic. The exact aperture values determined both via the heuristic and optimisation scheme can be found in Appendix B.

5 Discussion

Having obtained the necessary results, this section will delve into the interpretations, implications, and limitations of this study. Finally, a few recommendations are given for further research.

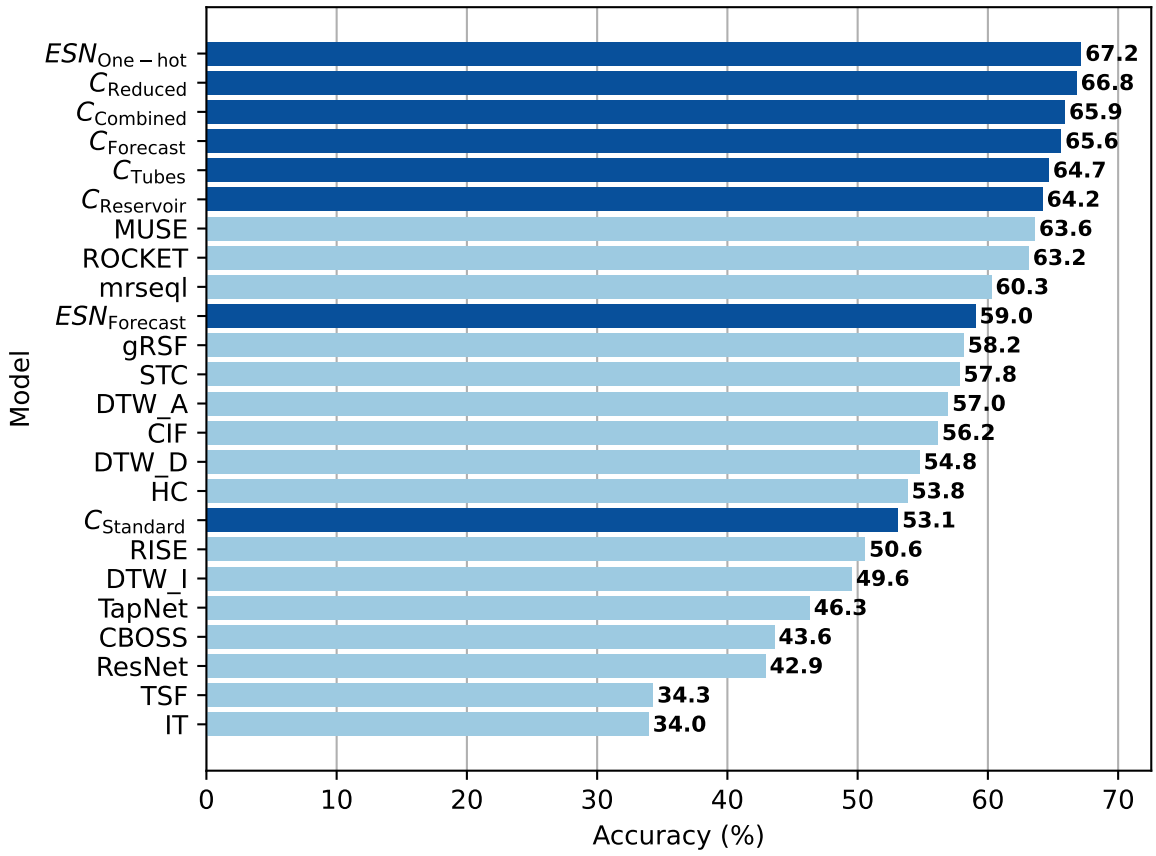

Figure 4.1: Accuracy of the conceptor and baseline classifiers compared to sixteen classifiers from previous work on the LSST dataset.

5.1 Interpretations & Implications

These results imply, first and foremost, that echo state networks are able to provide an information-rich enough feature representation through their non-linear expansion of its input, indicated by the relatively high performance of the baseline models compared to the classifiers from previous work.

Additionally, some conceptor classifiers such as $C_{Reduced}$, $C_{Combined}$, and $C_{Forecast}$ are roughly able to match the performance of the regularised linear classifier and able to outperform it when their aperture values are optimised.

Since the aperture values of the classifiers were optimised for a single randomly initialised underlying reservoir, quite large discrepancies in initial model performance using the aperture heuristic was observed. This might have led more "room" to optimise models with "bad" initialisations and less room to optimise models with "average" to "good" initialisations. While these results might imply that the $C_{Combined}$ classifier can benefit more from aperture adaptation than the other classifiers, more data is needed. It might be particularly useful to observe how models with good initial reservoirs respond to aperture optimisation. Additionally, from my observation (outside of the aperture analysis), it seems that optimal aperture values for the positive evidence conceptors can be found in a more narrow range across various random initialisations of the underlying network than the optimal aperture values for the negative evidence conceptors and are therefore more robust. This has driven the design decision to only optimize for a single reservoir ini-

tialisation.

While only being tested on a single dataset, these results also imply that conceptors can perform well on relatively complex non-stationary time series classification tasks and provide additional evidence that conceptor classifiers can perform competitively in these.

5.2 Limitations

There are, however, at least two points regarding possible limitations of this study that might provide additional nuance to the results.

First, the choice of performance metric was motivated by compatibility with the evaluations done on previous work provided in Ruiz et al. (2021). However, the average accuracy might be an uninstructive quality measure when the dataset is highly imbalanced as with the dataset in this study. Misclassifications for minority classes will only contribute a small amount to the final accuracy score. This means that a high score can still be achieved by only correctly classifying the majority classes.

Secondly, due to computational constraints, the search space concerning the size of the underlying reservoirs used for the optimisation scheme was set to an arbitrary size. In some cases, the most optimal solution was found for the largest permitted reservoir size. This implies that these models are possibly under-fitted, with no indication of the theoretical performance of bigger models. Therefore, a larger difference in performance between these models is likely to be observed when increasing the reservoir size to the point where the model might overfit on the training data. This issue had been partially addressed by assigning models of similar computational complexity an equal-sized search space for the reservoir size. However, one can better address this issue by searching over a space that is large enough to overfit the models.

5.3 Recommendations

The most straightforward and practical recommendation for further research would be to utilise these conceptor classifiers on a wide range of datasets; perhaps on all datasets presented in A. J. Bagnall et al. (2018). This will allow the overall performance of the presented classifiers to be more reliably estimated.

Additionally, following the previous section, given that the theoretical performance of some of the classifiers is most likely limited by the scale of the underlying ESNs, an optimisation scheme that includes larger ESN sizes will shed more light on absolute performance. This, in turn, will allow a more fair comparison with baseline models. An interesting way to explore this would be to use "diagonal conceptors" proposed in a Master's thesis by de Jong (2021). Instead of learning a conceptor per class with ${N_x}^2$ parameters as used in this research, diagonal matrices are used to construct conceptors, containing only N_x parameters. This cuts back on both computation time as well as memory usage quite significantly.

This, in turn, makes it interesting to extend the performance evaluation to also include an analysis on the computational complexity of the classifiers. As mentioned in the introduction, echo state networks and conceptors were conceived with, among other considerations, an eye for low computational burdens. It would be particularly interesting to compare processing times and memory requirements to state-of-the-art classifiers, especially deep-learning-based models.

Also in this light, since the hyperparameter μ is only utilised in a stage when the evidence values have already been computed and thus the models can be evaluated cheaply, it would be wise to optimise this parameter by sampling from its search space multiple times per run with the same evidence values. This will lessen the computation time as fewer full model evaluations will be needed to estimate μ reliably.

Moreover, it would be interesting to see how these conceptor classifiers perform on other types of classification tasks. As imagined hereafter, conceptors might lend themselves to be an intuitive tool for tasks which can be formulated in Boolean logic at a high-level.

One area of exploration could include tasks for which more conceptual "meta-data" is available. For example, hierarchical classification problems with taxonomic data, where the goal is to classify classes that are related to each other via a taxonomy. A possible solution to such problems using conceptors might be found in learning a conceptor for each class representing a "leaf node" in the taxonomy. Then, via Boolean logic, superclass conceptors can be created from the combination of leaf

node conceptors using the OR operation, effectively creating parent nodes. To classify a sample would then entail iteratively computing the evidence of each child node conceptor starting from the top of the hierarchy and selecting the node with the highest evidence. This process is then repeated with the child node now acting as the parent node until a leaf node is reached. This leaf node then corresponds to the classified class.

Another area that can be explored is multi-label classification, for which the goal is to assign multiple class labels to a sample. Qian, Zhang, and Wang (2019) have already explored this problem with the conceptor model proposed in Jaeger (2014) by computing additional conceptors that represent the data which are shared between multiple classes, using the AND operator on conceptors. These are then used to compute evidence values for signal characteristics belonging to all of the classes for which the conceptors were made.

Finally, it would be interesting to see by how much the performance of the classifiers $C_{\text{Reservoir}}$ and C_{Forecast}, which have their conceptors refer back to the ESN, can be improved by optimising their aperture values with an iterative optimisation scheme as used on the other conceptor classifiers in this research. Additionally, other (possibly cheaper) ways to optimise the aperture values should be explored. Using an evolutionary algorithm as in L. Wang, Wang, and Liu (2016) could be extended to also be used on the classifiers proposed in this paper.

References

Bagnall, A., Flynn, M., Large, J., Lines, J., & Middlehurst, M. (2020). On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (HIVE-COTE v1. 0). In *International Workshop on Advanced Analytics and Learning on Temporal Data* (pp. 3–18).

Bagnall, A. J., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., ... Keogh, E. J. (2018). The UEA multivariate time series classification archive, 2018. *CoRR*, *abs/1811.00075*. Retrieved from `http://arxiv.org/abs/1811.00075`

Bahmanyar, A., Biswas, R., Dai, M., Galbany, L., Hložek, R., Ishida, E. E. O., ... Collaboration, V. S. S. (2018). *The photometric LSST astronomical time-series classification challenge (PLaSTiCC): Data set.* Retrieved from `https://arxiv.org/abs/1810.00001`

de Jong, J. P. (2021). Controlling recurrent neural networks by diagonal conceptors. *CoRR*, *abs/2107.07968*. Retrieved from `https://arxiv.org/abs/2107.07968`

Dempster, A., Petitjean, F., & Webb, G. I. (2020). ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, *34*(5), 1454–1495.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., ... Petitjean, F. (2020). Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, *34*(6), 1936–1962.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, *148*.

Jaeger, H. (2014). Controlling recurrent neural networks by conceptors. *CoRR*, *abs/1403.3369*. Retrieved from `http://arxiv.org/abs/1403.3369`

Jaeger, H., Lukoševičius, M., Popovici, D., & Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, *20*(3), 335–352.

Karlsson, I., Papapetrou, P., & Boström, H. (2016). Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, *30*(5), 1053–1085.

Le Nguyen, T., Gsponer, S., Ilie, I., O'Reilly, M., & Ifrim, G. (2019). Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Mining and Knowledge Discovery*, *33*(4), 1183–1222.

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, *15*(2), 107–144.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019).

CATCH22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, *33*(6), 1821–1852.

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade* (p. 659-686). Springer.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Middlehurst, M., Large, J., & Bagnall, A. (2020). The canonical interval forest (CIF) classifier for time series classification. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 188–195).

Qian, G., Zhang, L., & Wang, Y. (2019). Single-label and multi-label conceptor classifiers in pre-trained neural networks. *Neural Computing and Applications*, *31*(10), 6179–6188.

Ruiz, A. P., Flynn, M., Large, J., Middlehurst, M., & Bagnall, A. (2021). The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *35*(2), 401–449.

Schäfer, P., & Högqvist, M. (2012). SFA: A symbolic Fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology* (pp. 516–527).

Schäfer, P., & Leser, U. (2017). Multivariate time series classification with WEASEL+MUSE. *CoRR*, *abs/1711.11343*. Retrieved from `http://arxiv.org/abs/1711.11343`

Shokoohi-Yekta, M., Hu, B., Jin, H., Wang, J., & Keogh, E. (2017). Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery*, *31*(1), 1–31.

Skowronski, M. D., & Harris, J. G. (2006). Minimum mean squared error time series classification using an echo state network prediction model. In *2006 IEEE International Symposium on Circuits and Systems* (pp. 3153–3156).

Skowronski, M. D., & Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, *20*(3), 414–423.

Wang, L., Wang, Z., & Liu, S. (2016). An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm. *Expert Systems with Applications*, *43*, 237–249.

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 1578–1585).

Ye, L., & Keogh, E. (2011). Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, *22*(1), 149–182.

Zhang, X., Gao, Y., Lin, J., & Lu, C.-T. (2020). Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, pp. 6845–6852).

A Model Parameters

Parameter	$C_{Standard}$	$C_{Reduced}$	$C_{Combined}$	C_{Tubes}	$C_{Reservoir}$	$C_{Forecast}$	$ESN_{One\text{-}hot}$	$ESN_{Forecast}$
ρ	1.06	1.07	1.23	1.30	0.67	0.67	1.33	1.57
$W^{\text{in}}_{\text{scale}}$	0.40	1.97	1.81	1.82	0.62	1.64	1.93	1.63
b_{scale}	0.40	0.91	1.90	1.40	0.71	0.91	1.77	1.04
a	0.28	0.38	0.39	0.10	0.17	0.17	0.73	0.15
N_x	25	500	500	80	85	100	500	420
μ	0.00	0.00	0.28	0.80	-	-	-	-
β	-	-	-	-	-	0.0001	0.18	0.076

Table A.1: Parameter values for the echo state network (ρ, $W^{\text{in}}_{\text{scale}}$, b_{scale}, a, and N_x), conceptors (μ), and Ridge regression (β).

B Aperture Values

	$C_{Standard}$		$C_{Reduced}$		$C_{Combined}$	
Conceptor	Heuristic	Optimised	Heuristic	Optimised	Heuristic	Optimised
C1	6.96	4.59	13.93	19.88	168.90	191.86
C2	51.98	77.93	59.71	68.73	90.51	105.98
C3	137.19	200.13	128.00	164.88	78.79	96.61
C4	222.86	143.29	168.90	205.19	68.59	96.51
C5	16.00	8.16	32.00	48.43	128.00	127.70
C6	1.52	1.23	1.52	2.08	103.97	85.87
C7	51.98	51.17	73.52	65.50	84.45	85.27
C8	6.06	3.24	18.38	14.76	274.37	150.99
C9	103.97	117.22	256.00	353.90	111.43	154.18
C10	17.15	14.50	42.22	32.12	128.00	146.63
C11	39.40	39.87	39.40	46.63	68.59	100.94
C12	1552.09	771.60	337.79	170.82	84.45	113.96
C13	22.63	18.44	36.758	25.09	97.01	89.31
C14	12.13	8.03	19.70	16.59	119.43	111.11

Table B.1: Heuristically determined and optimised aperture values of positive evidence conceptors. Depicted for classifiers $C_{Standard}$, $C_{Reduced}$, and $C_{Combined}$. Each conceptor represents a different class.

	$C_{Standard}$		$C_{Reduced}$		$C_{Combined}$	
Conceptor	Heuristic	Optimised	Heuristic	Optimised	Heuristic	Optimised
C1	1910.85	1950.21	181.02	177.27	64.00	71.23
C2	1910.85	1739.29	194.01	180.29	64.00	59.01
C3	2048.00	1934.67	194.01	200.33	64.00	58.52
C4	1910.85	1787.22	194.01	179.03	64.00	56.85
C5	1910.85	1983.06	181.02	190.54	59.71	65.20
C6	1910.85	1748.58	181.02	190.29	64.00	64.27
C7	1910.85	1723.30	181.02	176.76	64.00	60.80
C8	1910.85	2041.73	181.02	167.49	59.71	53.87
C9	1910.85	1795.24	181.02	172.61	59.71	52.27
C10	1910.85	1899.81	181.02	176.81	59.71	66.42
C11	1910.85	1991.44	194.01	206.76	64.00	60.53
C12	1782.89	1746.46	181.02	166.58	59.71	50.49
C13	1910.85	1737.00	194.01	177.59	64.00	56.69
C14	1910.85	1924.83	181.02	170.30	64.00	58.74

Table B.2: Heuristically determined and optimised aperture values of negative evidence conceptors. Depicted for classifiers $C_{Standard}$, $C_{Reduced}$, and $C_{Combined}$. Each conceptor represents a different class.